

[illegible][illegible]

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```

J 16  
16-Sep-1984 00:20:36  
5-Sep-1984 13:54:29

VAX-11 FORTRAN V3.4-56  
DISK\$VMSMASTER:[ERF.SRC]DUTUDRVR.FOR;1

Page 1

```
0001 C
0002 C Version:      'V04-000'
0003 C
0004 C*****
0005 C*
0006 C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0007 C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0008 C*  ALL RIGHTS RESERVED.
0009 C*
0010 C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0011 C*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0012 C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0013 C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0014 C*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0015 C*  TRANSFERRED.
0016 C*
0017 C*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0018 C*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0019 C*  CORPORATION.
0020 C*
0021 C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0022 C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0023 C*
0024 C*
0025 C*****
0026 C
0027
0028 c      Author  Brian Porter              Creation date  10-FEB-1982
0029
0030 c++
0031 c      Functional description:
0032 c
0033 c      This module displays entries logged by MSCP disks (dudriver) and
0034 c      and MSCP tapes (tudriver).
0035 c
0036 c      Modified by:
0037 c
0038 c      V03-007 EAD0200      Elliott A. Drayton      23-Jul-1984
0039 c      Added code to mark the begining of the intervening entries.
0040 c
0041 c      V03-006 SAR0272      Sharon A. Reynolds      18-Jun-1984
0042 c      - Re-structured and re-named the routines in this
0043 c      module to handle disk or tape MSCP entries for the
0044 c      addition of TMSCP support.
0045 c
0046 c      V03-005 SAR0197      Sharon A. Reynolds,      20-Feb-1984
0047 c      Added an SYE update that:
0048 c      - Removed 'invalid mscp command end message'.
0049 c
0050 c      V03-004 SAR0157      Sharon A. Reynolds,      12-Oct-1983
0051 c      Added an SYE update that:
0052 c      - adds an extra arguement to the 'dudriver_mscp_dispatcher'
0053 c      routine.
0054 c      - adds an extra arguement to the call for the
0055 c      'dudriver_mscp_dispatcher' routine.
0056 c      - adds an extra arguement to the calls for several
0057 c      routines that reside in 'mscp.for'.
```

K 16  
16-Sep-1984 00:20:36  
5-Sep-1984 13:54:29

VAX-11 FORTRAN V3.4-56  
DISK\$VMSMASTER:[ERF.SRC]DUTUDRIVR.FOR;1

Page 2

```
0058 C
0059 C      V03-003 SAR0072      Sharon A. Reynolds,      20-Jun-1983
0060 C      Changed the carriage control in the 'format' statements
0061 C      for use with ERF.
0062 C
0063 C      v03-002 BP0002      Brian Porter,      08-FEB-1983
0064 C      Corrected argument list to erllogmsg2.
0065 C
0066 C      v03-001 BP0001      Brian Porter,      19-APR-1982
0067 C      Made changes to accomodate invalid command mscp messages.
0068 C**
0069 C--
0070
0071 Subroutine DISK_TAPE_DRVR_MSCP_DISPATCHER (lun,option,recnt,
0072 1 mount_flag_and_label,record_length,queue_count)
0073
0074
0075 include 'src$:msghdr.for /nolist'
0134 include 'src$:emblmdef.for /nolist'
0203 include 'src$:embspdef.for /nolist'
0316
0317
0318 byte          lun
0319
0320 character*1    option
0321
0322 c This value RECCNT is not the record number of the entry just read from the
0323 c errlog.sys file it is the value which was saved in the queue when this
0324 c routine is called by DQ.
0325 integer*4      recnt
0326 c
0327 integer*4      mount_flag_and_label
0328 integer*4      record_length
0329 integer*4      queue_count
0330 integer*4      packet_length
0331
0332 byte          mslg$b_format
0333 equivalence    (emb(46),mslg$b_format)
0334
0335
0336 if (emb$w_hd_entry .eq. 100) then          ! Logmessage entry
0337
0338 C
0339 C Determine whether to output the long or short header and call
0340 C the appropriate routine.
0341 C
0342 If (queue_count .EQ. 1) then
0343
0344 Call FRCTOF (lun)
0345 Call HEADER2 (lun,recnt)
0346 Else
0347
0348 Call HEADER3 (lun,recnt)
0349 Endif
0350
0351 Call LOGGER (lun,'ERL$LOGMESSAGE ENTRY')
0352
```

```
0353      Call DHEAD3 (lun,'I/O',emb$b_lm_namlng,emb$t_lm_name,emb$w_lm_unit,  
0354      1 mount_flag_and_label)  
0355  
0356      Packet_length = record_length - 39  
0357  
0358      if (mslg$b_format .eq. 0) then          ! Controller error  
0359  
0360      if (option .eq. 'S') then  
0361      Call MSLG$K_CNT_ERR (lun,packet_length)  
0362      endif  
0363  
0364      else if (mslg$b_format .eq. 1) then      ! Memory access error  
0365  
0366      if (option .eq. 'S') then  
0367      Call MSLG$K_BUS_ADDR (lun,packet_length)  
0368      endif  
0369  
0370      else if (                               ! Disk/tape transfer error  
0371      1 mslg$b_format .eq. 2 ! mslg$k_disk_trn  
0372      1 .OR.  
0373      1 mslg$b_format .EQ. 5 ! mslg$k_tape_trn  
0374      1 ) then  
0375  
0376      if (option .eq. 'S') then  
0377      Call DISK_TAPE_TRANSFER_ERRORS (lun,packet_length)  
0378      endif  
0379  
0380      else if (                               ! SDI/STI errors  
0381      1 mslg$b_format .eq. 3 ! Disk SDI comm error - mslg$k_sdi  
0382      1 .OR.  
0383      1 mslg$b_format .EQ. 6 ! Tape STI comm or cmd failure - mslg$k_sti_err  
0384      1 .OR.  
0385      1 mslg$b_format .EQ. 7 ! Tape STI Drive Error Log - mslg$k_sti_del  
0386      1 .OR.  
0387      1 mslg$b_format .EQ. 8 ! Tape STI Formatter Error Log - mslg$k_sti_fel  
0388      1 ) then  
0389  
0390      if (option .eq. 'S') then  
0391      Call SDI_STI_ERRORS (lun,packet_length)  
0392      endif  
0393  
0394      else if (mslg$b_format .eq. 4) then      ! Small Disk error  
0395  
0396      if (option .eq. 'S') then  
0397      Call MSLG$K_SML_DSK (lun,packet_length)  
0398      endif  
0399  
0400      else  
0401      C  
0402      C Unknown format type, call a routine that will decode/output the header  
0403      C information and dump the rest of the packet in a hex longword format.  
0404      C  
0405      Call ERLLOGMSG2 (lun,record_length)  
0406      endif  
0407  
0408      else if (emb$w_hd_entry .eq. 99) then    ! Logstatus entry  
0409
```

```

0410      C
0411      C Determine whether to output the long or short header and call
0412      C the appropriate routine.
0413      C
0414          If (queue_count .EQ. 1) then
0415              Call FRCTOF (lun)
0416              Call HEADER2 (lun,recnt)
0417
0418          Else
0419              Call HEADER3 (lun,recnt)
0420          Endif
0421
0422          Call LOGGER (lun,'ERL$LOGSTATUS ENTRY')
0423
0424          Call DHEAD3 (lun,'I/O',emb$b_sp_namlng,emb$t_sp_name,emb$w_sp_unit,
0425                      1 mount_flag_and_label)
0426
0427          Call ERLLOGSTS2 (lun)
0428      endif
0429
0430      return
0431      end

```

## PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	426	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	45	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	164	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	1147	

## ENTRY POINTS

Address	Type	Name
0-00000000		DISK_TAPE_DRV_R_MSCP_DISPATCHER

## VARIABLES

Address	Type	Name	Address	Type	Name
3-00000010	L*1	EMBSB_LM_CLASS	3-00000014	L*1	EMBSB_LM_NAMLANG
3-00000011	L*1	EMBSB_LM_TYPE	3-00000010	L*1	EMBSB_SP_CLASS
3-00000040	L*1	EMBSB_SP_NAMLANG	3-00000011	L*1	EMBSB_SP_TYPE
3-00000000	I*4	EMBSL_HD_SID	3-00000014	I*4	EMBSL_SP_BCNT
3-00000038	I*4	EMBSL_SP_CHAR	3-0000003C	I*4	EMBSL_SP_CMDREF
3-00000020	I*4	EMBSL_SP_IOSB1	3-00000024	I*4	EMBSL_SP_IOSB2
3-00000018	I*4	EMBSL_SP_MEDIA	3-0000002C	I*4	EMBSL_SP_OPCNT
3-00000034	I*4	EMBSL_SP_OWNUIC	3-0000001C	I*4	EMBSL_SP_RQPID

3-00000015 CHAR EMB\$T\_LM\_NAME  
3-00000004 I\*2 EMB\$W\_HD\_ENTRY  
3-00000024 I\*2 EMB\$W\_LM\_MSGTYP  
3-00000012 I\*2 EMB\$W\_SP\_BOFF  
3-00000028 I\*2 EMB\$W\_SP\_FUNC  
3-0000002A I\*2 EMB\$W\_SP\_UNIT  
AP-00000010a I\*4 MOUNT\_FLAG\_AND\_LABEL  
AP-00000008a CHAR OPTION  
AP-00000018a I\*4 QUEUE\_COUNT  
AP-00000014a I\*4 RECORD\_LENGTH

3-00000041 CHAR EMB\$T\_SP\_NAME  
3-0000000E I\*2 EMB\$W\_HD\_ERRSEQ  
3-00000012 I\*2 EMB\$W\_LM\_UNIT  
3-00000030 I\*2 EMB\$W\_SP\_ERRCNT  
3-00000032 I\*2 EMB\$W\_SP\_STS  
AP-00000004a L\*1 LUN  
3-0000002E L\*1 MSLG\$B\_FORMAT  
2-00000000 I\*4 PACKET\_LENGTH  
AP-0000000Ca I\*4 RECCNT

## ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB\$B_LM_MSGTXT	460	(460)
3-00000006	I*4	EMB\$Q_HD_TIME	8	(2)

## FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	DHEAD3		DISK_TAPE_TRANSFER_ERRORS		ERLLOGMSG2
	ERLLOGSTS2		FRCTOF		HEADER2
	HEADER3		LOGGER		MSLG\$K_BUS_ADDR
	MSLG\$K_CNT_ERR		MSLG\$K_SML_DSK		SDI_STI_ERRORS

```

0001 c++
0002 c
0003 c
0004 c
0005 c Since mscp error log entries are delivered from the port via
0006 c the datagram service it is possible for them to be delivered
0007 c out of sequence or duplicated. It is the responsibility of
0008 c this queue to collect all entries containing the same command
0009 c reference for a given cpu together. They are placed in order
0010 c of error log entry type.
0011 c
0012 c The format of the elements are as follows
0013 c
0014 c +-----+
0015 c | flink1 |
0016 c +-----+
0017 c | blink1 |
0018 c +-----+
0019 c | logging sid |
0020 c +-----+
0021 c | root command reference flink |
0022 c +-----+
0023 c | root command reference blink |
0024 c +-----+
0025 c | command reference entry count |
0026 c +-----+
0027 c
0028 c +-----+
0029 c | flink2 |
0030 c +-----+
0031 c | blink2 |
0032 c +-----+
0033 c | command reference number |
0034 c +-----+
0035 c | root_emb$$$w_hd_entry_flink |
0036 c +-----+
0037 c | root_emb$$$w_hd_entry_blink |
0038 c +-----+
0039 c | emb$$$w_hd_entry count |
0040 c +-----+
0041 c
0042 c +-----+
0043 c | flink3 |
0044 c +-----+
0045 c | blink3 |
0046 c +-----+
0047 c | emb$$$w_hd_entry |
0048 c +-----+
0049 c | error log record number |
0050 c +-----+
0051 c | error log record size (bytes) |
0052 c +-----+
0053 c |
0054 c |-- --|
0055 c |
0056 c | error log record |
0057 c |

```

0058  
0059  
0060  
0061  
0062  
0063  
0064  
0065  
0066  
0067  
0068  
0069  
0070  
0071  
0072  
0073  
0074  
0075  
0076  
0077  
0136  
0205  
0318  
0319  
0320  
0321  
0322  
0323  
0324  
0325  
0326  
0327  
0328  
0329  
0330  
0331  
0332  
0333  
0334  
0335  
0336  
0337  
0338  
0339  
0340  
0341  
0342  
0343  
0344  
0345  
0346  
0347  
0348  
0349  
0350  
0351  
0352

```

c      +---+
c      |                                     |
c      |                                     |
c      |      mounted flag      |
c      |                                     |
c      |                                     |
c      |      mounted volume label (if any) |
c      |                                     |
c      +---+
c
c--

```

Subroutine DISK\_TAPE\_DRIVERS\_MSCP\_Q (record\_length,recnt,  
1 search\_command\_reference\_number)

include 'src\$:msghdr.for /nolist'  
include 'src\$:emblmdef.for /nolist'  
include 'src\$:embspdef.for /nolist'

byte	lun
character*1	option
integer*4	record_length
integer*4	recnt
integer*4	search_command_reference_number
integer*4	buffer(2)
logical*4	lib\$get_vm
integer*4	insert_blink
integer*4	logging_sid_entry_address
integer*4	command_reference_entry_address
integer*4	emb\$sw_hd_entry_address
integer*4	root_logging_sid_flink
integer*4	root_logging_sid_blink
equivalence	(buffer(1),root_logging_sid_flink)
equivalence	(buffer(2),root_logging_sid_blink)
integer*4	logging_sid_entry_count
data	logging_sid_entry_count /0/
integer*4	buffer1(6)
integer*4	flink1
integer*4	blink1
integer*4	logging_sid
integer*4	root_command_reference_flink
integer*4	root_command_reference_blink
integer*4	command_reference_entry_count
equivalence	(buffer1(1),flink1)
equivalence	(buffer1(2),blink1)

```
0353      equivalence      (buffer1(3),logging_sid)
0354      equivalence      (buffer1(4),root_command_reference_flink)
0355      equivalence      (buffer1(5),root_command_reference_blink)
0356      equivalence      (buffer1(6),command_reference_entry_count)
0357
0358      integer*4          buffer2(6)
0359      integer*4          flink2
0360      integer*4          blink2
0361      integer*4          command_reference_number
0362      integer*4          root_emb$$w_hd_entry_flink
0363      integer*4          root_emb$$w_hd_entry_blink
0364      integer*4          emb$$w_hd_entry_count
0365
0366      equivalence      (buffer2(1),flink2)
0367      equivalence      (buffer2(2),blink2)
0368      equivalence      (buffer2(3),command_reference_number)
0369      equivalence      (buffer2(4),root_emb$$w_hd_entry_flink)
0370      equivalence      (buffer2(5),root_emb$$w_hd_entry_blink)
0371      equivalence      (buffer2(6),emb$$w_hd_entry_count)
0372
0373      integer*4          buffer3(5)
0374      integer*4          flink3
0375      integer*4          blink3
0376      integer*4          emb$$w_hd_entry
0377      integer*4          error_log_record_number
0378      integer*4          error_log_record_length
0379
0380      equivalence      (buffer3(1),flink3)
0381      equivalence      (buffer3(2),blink3)
0382      equivalence      (buffer3(3),emb$$w_hd_entry)
0383      equivalence      (buffer3(4),error_log_record_number)
0384      equivalence      (buffer3(5),error_log_record_length)
0385
0386
0387      if (logging_sid_entry_count .eq. 0) then
0388
0389      root_logging_sid_flink = %loc(root_logging_sid_flink)
0390      root_logging_sid_blink = root_logging_sid_flink
0391      endif
0392
0393      logging_sid_entry_address = root_logging_sid_flink
0394
0395      do 100,i = 1,logging_sid_entry_count
0396
0397      call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)
0398
0399      if (logging_sid .eq. emb$l_hd_sid) then
0400
0401      10      command_reference_entry_address = root_command_reference_flink
0402
0403      do 90,j = 1,command_reference_entry_count
0404
0405      call movc3 (%val(24),%val(command_reference_entry_address),buffer2)
0406
0407      if (command_reference_number .eq. search_command_reference_number)
0408      1 then
0409
```

```
0410      25      insert_blink = root_emb$$w_hd_entry_blink
0411
0412      if (emb$$w_hd_entry_count .ne. 0) then
0413
0414      call movc3 (%val(12),%val(root_emb$$w_hd_entry_blink),buffer3)
0415
0416      if (emb$$w_hd_entry .lt. emb$w_hd_entry) then
0417
0418      insert_blink = blink3
0419      endif
0420      endif
0421
0422      call movc5 (%val(0),,%val(0),%val(20),buffer3)
0423
0424      if (lib$get_vm((20+record_length+16),emb$$w_hd_entry_address)) then
0425
0426      call insque (%val(emb$$w_hd_entry_address),%val(insert_blink))
0427
0428      emb$$w_hd_entry = emb$w_hd_entry
0429
0430      error_log_record_number = recnt
0431
0432      error_log_record_length = record_length
0433
0434      call movc3 (%val(12),emb$$w_hd_entry,
0435      1 %val(emb$$w_hd_entry_address + 8))
0436
0437      call movc3 (%val(record_length),emb,%val(emb$$w_hd_entry_address + 20))
0438
0439      call movl (-1,%val(emb$$w_hd_entry_address+20+record_length))
0440
0441      if (emb$w_hd_entry .eq. 100) then
0442
0443      call get_current_label (3,emb$l_hd_sid,emb$b_lm_namlng,emb$t_lm_name,
0444      1 emb$w_lm_unit,%val(emb$$w_hd_entry_address+20+record_length+4),*30)
0445
0446      else if (emb$w_hd_entry .eq. 99) then
0447
0448      call get_current_label (3,emb$l_hd_sid,emb$b_sp_namlng,emb$t_sp_name,
0449      1 emb$w_sp_unit,%val(emb$$w_hd_entry_address+20+record_length+4),*30)
0450      endif
0451
0452      call movl (emb$$w_hd_entry_address+20+record_length+4,
0453      1 %val(emb$$w_hd_entry_address+20+record_length))
0454
0455      30      emb$$w_hd_entry_count = emb$$w_hd_entry_count + 1
0456
0457      call movl (emb$$w_hd_entry_count,
0458      1 %val(command_reference_entry_address + 20))
0459      endif
0460
0461      return
0462      endif
0463
0464      command_reference_entry_address = flink2
0465
0466      90      continue
```

```
0467
0468      call movc5 (%val(0),,%val(0),%val(24),buffer2)
0469
0470      if (lib$get_vm(24,command_reference_entry_address)) then
0471
0472      call insque (%val(command_reference_entry_address),
0473      1 %val(root_command_reference_blink))
0474
0475      command_reference_number = search_command_reference_number
0476
0477      root_emb$$w_hd_entry_flink = command_reference_entry_address + 12
0478
0479      root_emb$$w_hd_entry_blink = root_emb$$w_hd_entry_flink
0480
0481      call movc3 (%val(16),command_reference_number,
0482      1 %val(command_reference_entry_address + 8))
0483
0484      command_reference_entry_count = command_reference_entry_count + 1
0485
0486      call movl (command_reference_entry_count,
0487      1 %val(logging_sid_entry_address + 20))
0488
0489      goto 25
0490    endif
0491
0492      return
0493    endif
0494
0495      logging_sid_entry_address = flink1
0496
0497 100    continue
0498
0499      call movc5 (%val(0),,%val(0),%val(24),buffer1)
0500
0501      if (lib$get_vm(24,logging_sid_entry_address)) then
0502
0503      call insque (%val(logging_sid_entry_address),
0504      1 %val(root_logging_sid_blink))
0505
0506      logging_sid = emb$l_hd_sid
0507
0508      root_command_reference_flink = logging_sid_entry_address + 12
0509
0510      root_command_reference_blink = root_command_reference_flink
0511
0512      call movc3 (%val(16),logging_sid,%val(logging_sid_entry_address + 8))
0513
0514      logging_sid_entry_count = logging_sid_entry_count + 1
0515
0516      goto 10
0517    endif
0518
0519      return
0520
0521
0522
0523      entry DISK_TAPE_DRIVERS_MSCP_DQ (lun,option)
```

```
0524
0525
0526      logging_sid_entry_address = root_logging_sid_flink
0527
0528      If (logging_sid_entry_count .GT. 0) then
0529          Write (lun, 9000)
0530      9000      Format (////////////////////
0531          1'      B E G I N I N G   O F   I N T E R V E N I N G   E N T R I E S')
0532      Endif
0533
0534      do 150,i = 1,logging_sid_entry_count
0535
0536          call movc3 (%val(24),%val(logging_sid_entry_address),buffer1)
0537
0538          command_reference_entry_address = root_command_reference_flink
0539
0540          do 200,j = 1,command_reference_entry_count
0541
0542              call movc3 (%val(24),%val(command_reference_entry_address),buffer2)
0543
0544              emb$$w_hd_entry_address = root_emb$$w_hd_entry_flink
0545
0546              do 250,k = 1,emb$$w_hd_entry_count
0547
0548                  call movc3 (%val(20),%val(emb$$w_hd_entry_address),buffer3)
0549
0550                  call movc5 (%val(0),,%val(0),%val(512),emb)
0551
0552                  call movc3 (%val(error_log_record_length),
0553                      1 %val(emb$$w_hd_entry_address + 20),emb)
0554
0555                  call DISK_TAPE_DRVR_MSCP_DISPATCHER (lun,option,
0556                      1 error_log_record_number,
0557                      1 %val(emb$$w_hd_entry_address+20+error_log_record_length),
0558                      1 error_log_record_length,k)
0559
0560              emb$$w_hd_entry_address = flink3
0561
0562      250      continue
0563
0564          command_reference_entry_address = flink2
0565
0566      200      continue
0567
0568          logging_sid_entry_address = flink1
0569
0570      150      continue
0571
0572      return
0573
0574      end
```

## PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	900	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	98	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	2058	

## ENTRY POINTS

Address	Type	Name	Address	Type	Name
0-00000290		DISK_TAPE_DRIVERS_MSCP_Q	0-00000000		DISK_TAPE_DRIVERS_MSCP_Q

## VARIABLES

Address	Type	Name	Address	Type	Name
2-00000030	I*4	BLINK1	2-00000018	I*4	BLINK2
2-00000004	I*4	BLINK3	2-00000054	I*4	COMMAND_REFERENCE_ENTRY_ADDRESS
2-00000040	I*4	COMMAND_REFERENCE_ENTRY_COUNT	2-0000001C	I*4	COMMAND_REFERENCE_NUMBER
2-00000008	I*4	EMB\$W_RD_ENTRY	2-00000058	I*4	EMB\$W_RD_ENTRY_ADDRESS
2-00000028	I*4	EMB\$W_HD_ENTRY_COUNT	3-00000010	L*1	EMB\$B_CM_CLASS
3-00000014	L*1	EMB\$B_CM_NAMLNG	3-00000011	L*1	EMB\$B_LM_TYPE
3-00000010	L*1	EMB\$B_SP_CLASS	3-00000040	L*1	EMB\$B_SP_NAMLNG
3-00000011	L*1	EMB\$B_SP_TYPE	3-00000000	I*4	EMB\$B_HD_SID
3-00000014	I*4	EMB\$B_SP_BCNT	3-00000038	I*4	EMB\$B_SP_CHAR
3-0000003C	I*4	EMB\$B_SP_CMDREF	3-00000020	I*4	EMB\$B_SP_IOSB1
3-00000024	I*4	EMB\$B_SP_IOSB2	3-00000018	I*4	EMB\$B_SP_MEDIA
3-0000002C	I*4	EMB\$B_SP_OPCNT	3-00000034	I*4	EMB\$B_SP_OWNUIC
3-0000001C	I*4	EMB\$B_SP_RQPID	3-00000015	CHAR	EMB\$T_LM_NAME
3-00000041	CHAR	EMB\$T_SP_NAME	3-00000004	I*2	EMB\$W_HD_ENTRY
3-0000000E	I*2	EMB\$W_HD_ERRSEQ	3-00000024	I*2	EMB\$W_LM_MSGTYP
3-00000012	I*2	EMB\$W_LM_UNIT	3-00000012	I*2	EMB\$W_SP_BOFF
3-00000030	I*2	EMB\$W_SP_ERRCNT	3-00000028	I*2	EMB\$W_SP_FUNC
3-00000032	I*2	EMB\$W_SP_STS	3-0000002A	I*2	EMB\$W_SP_UNIT
2-00000010	I*4	ERROR_LOG_RECORD_LENGTH	2-0000000C	I*4	ERROR_LOG_RECORD_NUMBER
2-0000002C	I*4	FLINK1	2-00000014	I*4	FLINK2
2-00000000	I*4	FLINK3	2-00000060	I*4	I
2-0000004C	I*4	INSERT_BLINK	2-00000064	I*4	J
2-00000068	I*4	K	2-00000034	I*4	LOGGING_SID
2-00000050	I*4	LOGGING_SID_ENTRY_ADDRESS	2-0000005C	I*4	LOGGING_SID_ENTRY_COUNT
AP-00000004a	L*1	LUN	AP-00000008a	CHAR	OPTION
AP-00000008a	I*4	RECCNT	AP-00000004a	I*4	RECORD_LENGTH
2-0000003C	I*4	ROOT_COMMAND_REFERENCE_BLINK	2-00000038	I*4	ROOT_COMMAND_REFERENCE_FLINK
2-00000024	I*4	ROOT_EMB\$W_RD_ENTRY_BLINK	2-00000020	I*4	ROOT_EMB\$W_RD_ENTRY_FLINK
2-00000048	I*4	ROOT_LOGGING_SID_BLINK	2-00000044	I*4	ROOT_LOGGING_SID_FLINK
AP-0000000Ca	I*4	SEARCH_COMMAND_REFERENCE_NUMBER			

## ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000044	I*4	BUFFER	8	(2)
2-0000002C	I*4	BUFFER1	24	(6)
2-00000014	I*4	BUFFER2	24	(6)
2-00000000	I*4	BUFFER3	20	(5)
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMBSB_LM_MSGTXT	460	(460)
3-00000006	I*4	EMBSQ_HD_TIME	8	(2)

## LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
0-0000004C	10	0-0000007D	25	0-0000019D	30	**	90	**	100	**	150
**	200	**	250	1-0000000C	9000'						

## FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
L*4	DISK_TAPE_DRV_R_MSCP_DISPATCHER		GET_CURRENT_LABEL		INSQUE
	LIB\$GET_VM		MOV C3		MOV C5
	MOVL				

```
0001
0002
0003
0004      Subroutine DUDRIVER_QIO (lun,emb$w_dv_func)
0005
0006
0007      include 'src$:qiocommon.for /nolist'
0008
0009
0010
0011
0012
0013
0014
0015
0016
0017
0018
0019
0020
0021
0022
0023
0024      byte          lun
0025
0026      integer*2      emb$w_dv_func
0027
0028      integer*4      qiocode(0:1,0:63)
0029
0030
0031
0032
0033
0034
0035
0036
0037
0038
0039
0040
0041
0042
0043
0044
0045
0046
0047
0048
0049
0050
0051
0052
0053
0054
0055
0056
0057
0058
0059
0060
0061
0062
0063
0064
0065
0066
0067
0068
0069
0070
0071
0072
0073
0074
0075
0076
0077
0078
0079
0080
0081
0082
0083
0084
0085
0086
0087
0088
0089
0090
0091
0092
0093
0094
0095
0096
0097
0098
0099
0100
0101
0102
0103
0104
0105
0106
0107
0108
0109
0110
0111
0112
0113
0114
0115
0116
0117
0118
0119
0120
0121
0122
0123
0124
0125
0126
0127
0128
0129
0130
0131
0132
0133
0134
0135
0136
0137
0138
0139
0140
0141
0142
0143
0144
0145
0146
0147
0148
0149
0150
0151
0152
0153
0154
0155
0156
0157
0158
0159
0160
0161
0162
0163
0164
0165
0166
0167
0168
0169
0170
0171
0172
0173
0174
0175
0176
0177
0178
0179
0180
0181
0182
0183
0184
0185
0186
0187
0188
0189
0190
0191
0192
0193
0194
0195
0196
0197
0198
0199
0200
0201
0202
0203
0204
0205
0206
0207
0208
0209
0210
0211
0212
0213
0214
0215
0216
0217
0218
0219
0220
0221
0222
0223
0224
0225
0226
0227
0228
0229
0230
0231
0232
0233
0234
0235
0236
0237
0238
0239
0240
0241
0242
0243
0244
0245
0246
0247
0248
0249
0250
0251
0252
0253
0254
0255
0256
0257
0258
0259
0260
0261
0262
0263
0264
0265
0266
0267
0268
0269
0270
0271
0272
0273
0274
0275
0276
0277
0278
0279
0280
0281
0282
0283
0284
0285
0286
0287
0288
0289
0290
0291
0292
0293
0294
0295
0296
0297
0298
0299
0300
0301
0302
0303
0304
0305
0306
0307
0308
0309
0310
0311
0312
0313
0314
0315
0316
0317
0318
0319
0320
```

```
0321      10      continue
0322      endif
0323
0324      call cdrp$w_func (lun,emb$w_dv_func,
0325      1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0326
0327      return
0328      end
```

## PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	243	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	2046	

## ENTRY POINTS

Address	Type	Name
0-00000000		DUDRIVER_QIO

## VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMB\$W_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IOS_ABORT	3-0000034D	CHAR	IOS_ACCESS
3-000003C2	CHAR	IOS_ACPCONTROL	3-000004B3	CHAR	IOS_AVAILABLE
3-00000297	CHAR	IOS_CLEAN	3-00000369	CHAR	IOS_CREATE
3-00000385	CHAR	IOS_DEACCESS	3-00000393	CHAR	IOS_DELETE
3-0000026D	CHAR	IOS_DIAGNOSE	3-00000065	CHAR	IOS_DRVCLR
3-000004CB	CHAR	IOS_DSE	3-000000A9	CHAR	IOS_ERASETAPE
3-00000276	CHAR	IOS_FORMAT	3-00000071	CHAR	IOS_INITIALIZE
3-00000014	CHAR	IOS_LOADMCODE	3-000003A1	CHAR	IOS_MODIFY
3-000003E2	CHAR	IOS_MOUNT	3-00000000	CHAR	IOS_NOP
3-0000009D	CHAR	IOS_OFFSET	3-000000EB	CHAR	IOS_PACKACK
3-000000E0	CHAR	IOS_QSTOP	3-000003EF	CHAR	IOS_RDSTATS
3-00000421	CHAR	IOS_READCSR	3-00000169	CHAR	IOS_READHEAD
3-000002B6	CHAR	IOS_READLBLK	3-0000013F	CHAR	IOS_READPBLK
3-00000200	CHAR	IOS_READPRESET	3-00000195	CHAR	IOS_READTRACKD
3-0000033A	CHAR	IOS_READVBLK	3-0000045A	CHAR	IOS_READWTHBUF
3-00000484	CHAR	IOS_READWTHXBUF	3-0000004D	CHAR	IOS_RECAL
3-0000007C	CHAR	IOS_RELEASE	3-000001AB	CHAR	IOS_REREADN
3-000001B8	CHAR	IOS_REREADP	3-000000CA	CHAR	IOS_RETCENTER
3-000002E6	CHAR	IOS_REWIND	3-000002C9	CHAR	IOS_REWINDOFF
3-000000FC	CHAR	IOS_SEARCH	3-00000024	CHAR	IOS_SEEK
3-00000231	CHAR	IOS_SENSECHAR	3-00000309	CHAR	IOS_SENSEMODE

DUDRIVER\_QIO

M 1  
16-Sep-1984 00:20:36  
5-Sep-1984 13:54:29

VAX-11 FORTRAN V3.4-56 Page 16  
DISK\$VMSMASTER:[ERF.SRC]DUTUDRIVR.FOR;1

3-0000021D CHAR IOS\_SETCHAR  
3-00000088 CHAR IOS\_SETCLOCKP  
3-000002ED CHAR IOS\_SKIPFILE  
3-00000029 CHAR IOS\_SPACEFILE  
3-000003D7 CHAR IOS\_STARTDATA  
3-00000037 CHAR IOS\_STARTMPROC  
3-00000059 CHAR IOS\_STOP  
3-00000468 CHAR IOS\_WRITEBUFNCRC  
3-000001E4 CHAR IOS\_WRITECHECKH  
3-00000153 CHAR IOS\_WRITEHEAD  
3-00000247 CHAR IOS\_WRITEMARK  
3-0000012A CHAR IOS\_WRITEPBLK  
3-0000017E CHAR IOS\_WRIETTRACKD  
3-00000448 CHAR IOS\_WRITEWITHBUF  
AP-00000004a L\*1 LUN

3-000003B8 CHAR IOS\_SETCLOCK  
3-000002DD CHAR IOS\_SETMODE  
3-000002FA CHAR IOS\_SKIPRECORD  
3-0000010E CHAR IOS\_SPACERECORD  
3-000000B4 CHAR IOS\_STARTDATAP  
3-0000020F CHAR IOS\_STARTSPNDL  
3-0000000D CHAR IOS\_UNLOAD  
3-0000011E CHAR IOS\_WRITECHECK  
3-000003FF CHAR IOS\_WRITECSR  
3-000002A2 CHAR IOS\_WRITEBLK  
3-00000314 CHAR IOS\_WRITEOF  
3-000001C9 CHAR IOS\_WRITERET  
3-00000326 CHAR IOS\_WRITEVBLK  
3-00000257 CHAR IOS\_WRTTMKR  
3-000004A1 CHAR QIO\_STRING

#### ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	I*4	QIOCODE	512	(0:1, 0:63)

#### LABELS

Address	Label
**	10

#### FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	CDRP\$W_FUNC	I*4	LIB\$EXTZV

```
0001
0002
0003
0004      Subroutine TUDRIVER_QIO (lun,emb$w_dv_func)
0005
0006      include 'src$:qiocommon.for /nolist'
0270
0271
0272      byte          lun
0273
0274      integer*2      emb$w_dv_func
0275
0276      integer*4      qiocode(0:1,0:63)
0277
0278
0279
0280      if (qiocode(0,0) .eq. 0) then
0281
0282      qiocode(1,00) = %loc(io$_nop)
0283      qiocode(1,01) = %loc(io$_unload)
0284      qiocode(1,08) = %loc(io$_packack)
0285
0286      qiocode(1,10) = %loc(io$_writecheck)
0287      qiocode(1,11) = %loc(io$_writepblk)
0288      qiocode(1,12) = %loc(io$_readpblk)
0289
0290      qiocode(1,17) = %loc(io$_available)
0291      qiocode(1,21) = %loc(io$_dse)
0292      qiocode(1,26) = %loc(io$_setchar)
0293
0294      qiocode(1,27) = %loc(io$_sensechar)
0295      qiocode(1,32) = %loc(io$_writelblk)
0296      qiocode(1,33) = %loc(io$_readlblk)
0297
0298      qiocode(1,35) = %loc(io$_setmode)
0299      qiocode(1,39) = %loc(io$_sensemode)
0300      qiocode(1,48) = %loc(io$_writevblk)
0301
0302      qiocode(1,49) = %loc(io$_readvblk)
0303      qiocode(1,50) = %loc(io$_access)
0304      qiocode(1,51) = %loc(io$_create)
0305
0306      qiocode(1,52) = %loc(io$_deaccess)
0307      qiocode(1,53) = %loc(io$_delete)
0308      qiocode(1,54) = %loc(io$_modify)
0309
0310      qiocode(1,56) = %loc(io$_acpcontrol)
0311      qiocode(1,57) = %loc(io$_mount)
0312
0313      do 10,i = 0,63
0314
0315      qiocode(0,i) = 33
0316
0317      if (qiocode(1,i) .eq. 0) then
0318      qiocode(1,i) = %loc(qio_string)
0319      endif
0320
```

```

0321      10      continue
0322      endif
0323
0324      call cdrp$w_func (lun,emb$w_dv_func,
0325      1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0326
0327      return
0328      end

```

## PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	250	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	2053	

## ENTRY POINTS

Address	Type	Name
0-00000000		TUDRIVER_QIO

## VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMB\$W_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IOS_ABORT	3-0000034D	CHAR	IOS_ACCESS
3-000003C2	CHAR	IOS_ACPCONTROL	3-000004B3	CHAR	IOS_AVAILABLE
3-00000297	CHAR	IOS_CLEAN	3-00000369	CHAR	IOS_CREATE
3-00000385	CHAR	IOS_DEACCESS	3-00000393	CHAR	IOS_DELETE
3-0000026D	CHAR	IOS_DIAGNOSE	3-00000065	CHAR	IOS_DRVCLR
3-000004CB	CHAR	IOS_DSE	3-000000A9	CHAR	IOS_ERASETAPE
3-00000276	CHAR	IOS_FORMAT	3-00000071	CHAR	IOS_INITIALIZE
3-00000014	CHAR	IOS_LOADMCODE	3-000003A1	CHAR	IOS_MODIFY
3-000003E2	CHAR	IOS_MOUNT	3-00000000	CHAR	IOS_NOP
3-0000009D	CHAR	IOS_OFFSET	3-000000EB	CHAR	IOS_PACKACK
3-000000E0	CHAR	IOS_QSTOP	3-000003EF	CHAR	IOS_RDSTATS
3-00000421	CHAR	IOS_READCSR	3-00000169	CHAR	IOS_READHEAD
3-000002B6	CHAR	IOS_READLBLK	3-0000013F	CHAR	IOS_READPBLK
3-00000200	CHAR	IOS_READPRESET	3-00000195	CHAR	IOS_READTRACKD
3-0000033A	CHAR	IOS_READVBLK	3-0000045A	CHAR	IOS_READWTHBUF
3-00000484	CHAR	IOS_READWTHXBUF	3-0000004D	CHAR	IOS_RECAL
3-0000007C	CHAR	IOS_RELEASE	3-000001AB	CHAR	IOS_REREADN
3-000001B8	CHAR	IOS_REREADP	3-000000CA	CHAR	IOS_RETCENTER
3-000002E6	CHAR	IOS_REWIND	3-000002C9	CHAR	IOS_REWINDOFF
3-000000FC	CHAR	IOS_SEARCH	3-00000024	CHAR	IOS_SEEK
3-00000231	CHAR	IOS_SENSECHAR	3-00000309	CHAR	IOS_SENSEMODE

TUDRIVER\_QIO

C 2  
16-Sep-1984 00:20:36  
5-Sep-1984 13:54:29

VAX-11 FORTRAN V3.4-56 Page 19  
DISK\$VMSMASTER:[ERF.SRC]DUTUDRIVR.FOR;1

3-0000021D	CHAR	IOS_SETCHAR	3-000003B8	CHAR	IOS_SETCLOCK
3-00000088	CHAR	IOS_SETCLOCKP	3-000002DD	CHAR	IOS_SETMODE
3-000002ED	CHAR	IOS_SKIPFILE	3-000002FA	CHAR	IOS_SKIPRECORD
3-00000029	CHAR	IOS_SPACEFILE	3-0000010E	CHAR	IOS_SPACERECORD
3-000003D7	CHAR	IOS_STARTDATA	3-000000B4	CHAR	IOS_STARTDATAP
3-00000037	CHAR	IOS_STARTMPROC	3-0000020F	CHAR	IOS_STARTSPNDL
3-00000059	CHAR	IOS_STOP	3-0000000D	CHAR	IOS_UNLOAD
3-0000046B	CHAR	IOS_WRITEBUFNCRC	3-0000011E	CHAR	IOS_WRITECHECK
3-000001E4	CHAR	IOS_WRITECHECKH	3-000003FF	CHAR	IOS_WRITECSR
3-00000153	CHAR	IOS_WRITEHEAD	3-000002A2	CHAR	IOS_WRITEBLK
3-00000247	CHAR	IOS_WRITEMARK	3-00000314	CHAR	IOS_WRITEOF
3-0000012A	CHAR	IOS_WRITEPBLK	3-000001C9	CHAR	IOS_WRITERET
3-0000017E	CHAR	IOS_WRITETRACKD	3-00000326	CHAR	IOS_WRITEVBLK
3-00000448	CHAR	IOS_WRITEWITHBUF	3-00000257	CHAR	IOS_WRTTMKR
AP-00000004a	L*1	LUN	3-000004A1	CHAR	QIO_STRING

#### ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	I*4	QIOCODE	512	(0:1, 0:63)

#### LABELS

Address	Label
**	10

#### FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	CDRPSW_FUNC	I*4	LIB\$EXTZV

#### COMMAND QUALIFIERS

FORTRAN /LIS=LISS:DUTUDRIVR/OBJ=OBJ\$:DUTUDRIVR MSRC\$:DUTUDRIVR

/CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE\_FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)

/F77 /NOG\_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD\_LINES /NOCROSS\_REFERENCE /NOMACHINE\_CODE /CONTINUATIONS=19

#### COMPILATION STATISTICS

Run Time:	11.00 seconds
Elapsed Time:	28.53 seconds
Page Faults:	373
Dynamic Memory:	199 pages

0147

AH-BT13A-SE  
VAX/VMS V4.0

**DIGITAL EQUIPMENT CORPORATION**  
**CONFIDENTIAL AND PROPRIETARY**

ERF1	ERF2	ERF3	ERF4	ERF5	ERF6	ERF7	ERF8	ERF9	ERF10	ERF11	ERF12	ERF13	ERF14	ERF15	ERF16	ERF17	ERF18	ERF19	ERF20	ERF21	ERF22	ERF23	ERF24	ERF25	ERF26	ERF27	ERF28	ERF29	ERF30	ERF31	ERF32	ERF33	ERF34	ERF35	ERF36	ERF37	ERF38	ERF39	ERF40	ERF41	ERF42	ERF43	ERF44	ERF45	ERF46	ERF47	ERF48	ERF49	ERF50	ERF51	ERF52	ERF53	ERF54	ERF55	ERF56	ERF57	ERF58	ERF59	ERF60	ERF61	ERF62	ERF63	ERF64	ERF65	ERF66	ERF67	ERF68	ERF69	ERF70	ERF71	ERF72	ERF73	ERF74	ERF75	ERF76	ERF77	ERF78	ERF79	ERF80	ERF81	ERF82	ERF83	ERF84	ERF85	ERF86	ERF87	ERF88	ERF89	ERF90	ERF91	ERF92	ERF93	ERF94	ERF95	ERF96	ERF97	ERF98	ERF99	ERF100
ERF101	ERF102	ERF103	ERF104	ERF105	ERF106	ERF107	ERF108	ERF109	ERF110	ERF111	ERF112	ERF113	ERF114	ERF115	ERF116	ERF117	ERF118	ERF119	ERF120	ERF121	ERF122	ERF123	ERF124	ERF125	ERF126	ERF127	ERF128	ERF129	ERF130	ERF131	ERF132	ERF133	ERF134	ERF135	ERF136	ERF137	ERF138	ERF139	ERF140	ERF141	ERF142	ERF143	ERF144	ERF145	ERF146	ERF147	ERF148	ERF149	ERF150	ERF151	ERF152	ERF153	ERF154	ERF155	ERF156	ERF157	ERF158	ERF159	ERF160	ERF161	ERF162	ERF163	ERF164	ERF165	ERF166	ERF167	ERF168	ERF169	ERF170	ERF171	ERF172	ERF173	ERF174	ERF175	ERF176	ERF177	ERF178	ERF179	ERF180	ERF181	ERF182	ERF183	ERF184	ERF185	ERF186	ERF187	ERF188	ERF189	ERF190	ERF191	ERF192	ERF193	ERF194	ERF195	ERF196	ERF197	ERF198	ERF199	ERF200
ERF201	ERF202	ERF203	ERF204	ERF205	ERF206	ERF207	ERF208	ERF209	ERF210	ERF211	ERF212	ERF213	ERF214	ERF215	ERF216	ERF217	ERF218	ERF219	ERF220	ERF221	ERF222	ERF223	ERF224	ERF225	ERF226	ERF227	ERF228	ERF229	ERF230	ERF231	ERF232	ERF233	ERF234	ERF235	ERF236	ERF237	ERF238	ERF239	ERF240	ERF241	ERF242	ERF243	ERF244	ERF245	ERF246	ERF247	ERF248	ERF249	ERF250	ERF251	ERF252	ERF253	ERF254	ERF255	ERF256	ERF257	ERF258	ERF259	ERF260	ERF261	ERF262	ERF263	ERF264	ERF265	ERF266	ERF267	ERF268	ERF269	ERF270	ERF271	ERF272	ERF273	ERF274	ERF275	ERF276	ERF277	ERF278	ERF279	ERF280	ERF281	ERF282	ERF283	ERF284	ERF285	ERF286	ERF287	ERF288	ERF289	ERF290	ERF291	ERF292	ERF293	ERF294	ERF295	ERF296	ERF297	ERF298	ERF299	ERF300
ERF301	ERF302	ERF303	ERF304	ERF305	ERF306	ERF307	ERF308	ERF309	ERF310	ERF311	ERF312	ERF313	ERF314	ERF315	ERF316	ERF317	ERF318	ERF319	ERF320	ERF321	ERF322	ERF323	ERF324	ERF325	ERF326	ERF327	ERF328	ERF329	ERF330	ERF331	ERF332	ERF333	ERF334	ERF335	ERF336	ERF337	ERF338	ERF339	ERF340	ERF341	ERF342	ERF343	ERF344	ERF345	ERF346	ERF347	ERF348	ERF349	ERF350	ERF351	ERF352	ERF353	ERF354	ERF355	ERF356	ERF357	ERF358	ERF359	ERF360	ERF361	ERF362	ERF363	ERF364	ERF365	ERF366	ERF367	ERF368	ERF369	ERF370	ERF371	ERF372	ERF373	ERF374	ERF375	ERF376	ERF377	ERF378	ERF379	ERF380	ERF381	ERF382	ERF383	ERF384	ERF385	ERF386	ERF387	ERF388	ERF389	ERF390	ERF391	ERF392	ERF393	ERF394	ERF395	ERF396	ERF397	ERF398	ERF399	ERF400
ERF401	ERF402	ERF403	ERF404	ERF405	ERF406	ERF407	ERF408	ERF409	ERF410	ERF411	ERF412	ERF413	ERF414	ERF415	ERF416	ERF417	ERF418	ERF419	ERF420	ERF421	ERF422	ERF423	ERF424	ERF425	ERF426	ERF427	ERF428	ERF429	ERF430	ERF431	ERF432	ERF433	ERF434	ERF435	ERF436	ERF437	ERF438	ERF439	ERF440	ERF441	ERF442	ERF443	ERF444	ERF445	ERF446	ERF447	ERF448	ERF449	ERF450	ERF451	ERF452	ERF453	ERF454	ERF455	ERF456	ERF457	ERF458	ERF459	ERF460	ERF461	ERF462	ERF463	ERF464	ERF465	ERF466	ERF467	ERF468	ERF469	ERF470	ERF471	ERF472	ERF473	ERF474	ERF475	ERF476	ERF477	ERF478	ERF479	ERF480	ERF481	ERF482	ERF483	ERF484	ERF485	ERF486	ERF487	ERF488	ERF489	ERF490	ERF491	ERF492	ERF493	ERF494	ERF495	ERF496	ERF497	ERF498	ERF499	ERF500
ERF501	ERF502	ERF503	ERF504	ERF505	ERF506	ERF507	ERF508	ERF509	ERF510	ERF511	ERF512	ERF513	ERF514	ERF515	ERF516	ERF517	ERF518	ERF519	ERF520	ERF521	ERF522	ERF523	ERF524	ERF525	ERF526	ERF527	ERF528	ERF529	ERF530	ERF531	ERF532	ERF533	ERF534	ERF535	ERF536	ERF537	ERF538	ERF539	ERF540	ERF541	ERF542	ERF543	ERF544	ERF545	ERF546	ERF547	ERF548	ERF549	ERF550	ERF551	ERF552	ERF553	ERF554	ERF555	ERF556	ERF557	ERF558	ERF559	ERF560	ERF561	ERF562	ERF563	ERF564	ERF565	ERF566	ERF567	ERF568	ERF569	ERF570	ERF571	ERF572	ERF573	ERF574	ERF575	ERF576	ERF577	ERF578	ERF579	ERF580	ERF581	ERF582	ERF583	ERF584	ERF585	ERF586	ERF587	ERF588	ERF589	ERF590	ERF591	ERF592	ERF593	ERF594	ERF595	ERF596	ERF597	ERF598	ERF599	ERF600
ERF601	ERF602	ERF603	ERF604	ERF605	ERF606	ERF607	ERF608	ERF609	ERF610	ERF611	ERF612	ERF613	ERF614	ERF615	ERF616	ERF617	ERF618	ERF619	ERF620	ERF621	ERF622	ERF623	ERF624	ERF625	ERF626	ERF627	ERF628	ERF629	ERF630	ERF631	ERF632	ERF633	ERF634	ERF635	ERF636	ERF637	ERF638	ERF639	ERF640	ERF641	ERF642	ERF643	ERF644	ERF645	ERF646	ERF647	ERF648	ERF649	ERF650	ERF651	ERF652	ERF653	ERF654	ERF655	ERF656	ERF657	ERF658	ERF659	ERF660	ERF661	ERF662	ERF663	ERF664	ERF665	ERF666	ERF667	ERF668	ERF669	ERF670	ERF671	ERF672	ERF673	ERF674	ERF675	ERF676	ERF677	ERF678	ERF679	ERF680	ERF681	ERF682	ERF683	ERF684	ERF685	ERF686	ERF687	ERF688	ERF689	ERF690	ERF691	ERF692	ERF693	ERF694	ERF695	ERF696	ERF697	ERF698	ERF699	ERF700
ERF701	ERF702	ERF703	ERF704	ERF705	ERF706	ERF707	ERF708	ERF709	ERF710	ERF711	ERF712	ERF713	ERF714	ERF715	ERF716	ERF717	ERF718	ERF719	ERF720	ERF721	ERF722	ERF723	ERF724	ERF725	ERF726	ERF727	ERF728	ERF729	ERF730	ERF731	ERF732	ERF733	ERF734	ERF735	ERF736	ERF737	ERF738	ERF739	ERF740	ERF741	ERF742	ERF743	ERF744	ERF745	ERF746	ERF747	ERF748	ERF749	ERF750	ERF751	ERF752	ERF753	ERF754	ERF755	ERF756	ERF757	ERF758	ERF759	ERF760	ERF761	ERF762	ERF763	ERF764	ERF765	ERF766	ERF767	ERF768	ERF769	ERF770	ERF771	ERF772	ERF773	ERF774	ERF775	ERF776	ERF777	ERF778	ERF779	ERF780	ERF781	ERF782	ERF783	ERF784	ERF785	ERF786	ERF787	ERF788	ERF789	ERF790	ERF791	ERF792	ERF793	ERF794	ERF795	ERF796	ERF797	ERF798	ERF799	ERF800
ERF801	ERF802	ERF803	ERF804	ERF805	ERF806	ERF807	ERF808	ERF809	ERF810	ERF811	ERF812	ERF813	ERF814	ERF815	ERF816	ERF817	ERF818	ERF819	ERF820	ERF821	ERF822	ERF823	ERF824	ERF825	ERF826	ERF827	ERF828	ERF829	ERF830	ERF831	ERF832	ERF833	ERF834	ERF835	ERF836	ERF837	ERF838	ERF839	ERF840	ERF841	ERF842	ERF843	ERF844	ERF845	ERF846	ERF847	ERF848	ERF849	ERF850	ERF851	ERF852	ERF853	ERF854	ERF855	ERF856	ERF857	ERF858	ERF859	ERF860	ERF861	ERF862	ERF863	ERF864	ERF865	ERF866	ERF867	ERF868	ERF869	ERF870	ERF871	ERF872	ERF873	ERF874	ERF875	ERF876	ERF877	ERF878	ERF879	ERF880	ERF881	ERF882	ERF883	ERF884	ERF885	ERF886	ERF887	ERF888	ERF889	ERF890	ERF891	ERF892	ERF893	ERF894	ERF895	ERF896	ERF897	ERF898	ERF899	ERF900
ERF901	ERF902	ERF903	ERF904	ERF905	ERF906	ERF907	ERF908	ERF909	ERF910	ERF911	ERF912	ERF913	ERF914	ERF915	ERF916	ERF917	ERF918	ERF919	ERF920	ERF921	ERF922	ERF923	ERF924	ERF925	ERF926	ERF927	ERF928	ERF929	ERF930	ERF931	ERF932	ERF933	ERF934	ERF935	ERF936	ERF937	ERF938	ERF939	ERF940	ERF941	ERF942	ERF943	ERF944	ERF945	ERF946	ERF947	ERF948	ERF949	ERF950	ERF951	ERF952	ERF953	ERF954	ERF955	ERF956	ERF957	ERF958	ERF959	ERF960	ERF961	ERF962	ERF963	ERF964	ERF965	ERF966	ERF967	ERF968	ERF969	ERF970	ERF971	ERF972	ERF973	ERF974	ERF975	ERF976	ERF977	ERF978	ERF979	ERF980	ERF981	ERF982	ERF983	ERF984	ERF985	ERF986	ERF987	ERF988	ERF989	ERF990	ERF991	ERF992	ERF993	ERF994	ERF995	ERF996	ERF997	ERF998	ERF999	ERF1000